

5.2 Transport layer - NETBLT. The bulk data transfer protocol NETBLT forms the transport layer of the TACO2 protocol suite. NETBLT in TACO2 works by opening a connection between a sender and a receiver (at the sender's request), transferring a message in one or more buffers, and closing the connection. Each buffer is transferred as a sequence of packets; the interaction between sender and receiver is primarily on a per-buffer basis. An overview of NETBLT is provided in 5.2.1; further explanation and detailed requirements are found in the following sections. The material here assumes the existence of a full-duplex connection between sender and receiver, where packets can be transferred in both directions concurrently. Changes for half-duplex and simplex cases are provided in 5.2.7. Specific packet types are identified in the following sections by upper-case names (DATA packets), in contrast with packet functions (keepalive packets), which are accomplished by more than one packet type.

5.2.1 NETBLT full-duplex operations. NETBLT opens a connection between two "clients" (the "sender" and the "receiver"), transfers the data in a series of large data aggregates called "buffers," and then closes the connection. (In TACO2, those clients are the TACO2 NRTS layer). NETBLT transfers each buffer as a sequence of packets.

5.2.1.1 Single-buffer operation. In its simplest form, a NETBLT transfer works as follows: the sending client loads a buffer of data and passes it to the NETBLT layer to be transferred. NETBLT breaks the buffer up into packets and sends these packets across the network via IP datagrams. The receiving NETBLT loads these packets into a matching buffer. When the last packet in the buffer should have arrived at the receiver, the receiving NETBLT checks whether all packets in that buffer have been received correctly. If some packets have not been received correctly, the receiving NETBLT requests that they be resent. When the buffer has been received completely, the receiving NETBLT passes it to the receiving client. When a new buffer is ready to receive more data, the receiving NETBLT notifies the sender that the new buffer is ready, and the sender prepares and sends the next buffer in the same manner. This continues until all the data has been sent; at that time the sender notifies the receiver that the transmission has been completed. The two sides then close the connection.

5.2.1.2 Multiple-buffer operation. As described in 5.2.1.1, the NETBLT protocol is "lock-step." Action halts after a buffer is transmitted, and begins again after confirmation is received from the receiver of data. NETBLT provides for multiple buffering, so that the sending NETBLT can transmit new buffers while earlier buffers are waiting for confirmation from the receiving NETBLT.

## 5.2.2 Buffers and packets.

5.2.2.1 Buffers. The data to be transmitted shall be broken into buffers by the sending client. Each buffer shall be the same size, except for the last buffer. During connection setup, the sending and receiving NETBLTs shall negotiate the buffer size. Buffer sizes shall be in bytes only; data shall be placed in buffers on byte boundaries.

5.2.2.2 Packets. Buffers are broken down by NETBLT into sequences of DATA packets. DATA packet size shall be negotiated between the sending and receiving NETBLTs during

connection setup. All DATA packets shall be the same size. The last packet of every buffer is not a DATA packet but rather an LDATA packet. DATA and LDATA packets are identical in format except for the packet type.

5.2.3 Flow control. NETBLT uses two strategies for flow control, one at the client level and one internal.

5.2.3.1 Client level flow control. The sending and receiving NETBLTs transmit data in buffers; therefore, client flow control is by buffer. Before a buffer can be transmitted, NETBLT confirms that both clients have set up matching buffers, that one is ready to send data, and that the other is ready to receive data. Either client can control data flow by not providing a new buffer. Clients cannot stop a buffer transfer once it is in progress, except by aborting the entire transfer.

5.2.3.2 Internal flow control. The internal flow control mechanism for NETBLT is rate control. The transmission rate is negotiated by the sending and receiving NETBLTs during connection setup and after each buffer transmission. The sender uses timers to maintain the negotiated rate, by controlling the time to transmit groups of packets. The sender transmits a burst of packets over the negotiated time interval, and sends another burst in the next interval, therefore, NETBLT's rate control has two parts, a burst size and a burst interval, with  $(\text{burst interval})/(\text{burst size})$  equal to the average transmission time per packet.

5.2.3.3 Flow control parameter negotiation. All NETBLT flow control parameters (packet size, buffer size, number of buffers outstanding, burst size, and burst interval) shall be negotiated during connection setup. The negotiation process is the same for all parameters. The client initiating the connection (the sender) shall send a value for each parameter in its OPEN packet. The other client (the receiver) shall compare these values with the highest-performance values it can support. The receiver can modify any of the parameters, but only by making them more restrictive, such as, smaller packet size, smaller buffer size, fewer buffers, smaller burst size, and larger burst interval. The (possibly modified) parameters shall be sent back to the sender in the RESPONSE packet.

5.2.3.4 Flow control parameter renegotiation. The burst size and burst interval also may be re-negotiated after each buffer transmission to adjust the transfer rate according to the performance observed from transferring the previous buffer. The receiving end shall send burst size and burst interval values in its OK messages (described in 5.2.5.2.1). The sender shall compare these values with the values it can support. It then may modify either of the parameters, but only by making them more restrictive. The modified parameters shall then be communicated to the receiver in DATA, LDATA, or NULL-ACK packets.

5.2.3.5 Client-controlled flow. (Effectivity 3). A burst interval value of zero shall have a special meaning: internal flow control shall be turned off, so that only client level flow control shall be in effect. In this case, the sending NETBLT shall transmit packets without regard for the rate control mechanism. When using client-controlled flow, the receiver shall use an alternative method for data timer estimation (see 5.2.5.2.4.3).

5.2.4 Checksumming. NETBLT shall use checksums to validate the contents of packets and packet headers unless packet integrity is assured by the data link layer. The checksum value shall be

the bitwise negation of the ones-complement sum of the 16-bit words being checksummed. On twos-complement machines, the ones-complement sum can be computed by means of an "end around carry"; that is, any overflows from the most significant bit are added into the least significant bits. See figure 8 for an example. If the quantity to be checksummed has an odd number of bytes, it shall be padded with a final null byte (binary 0's) to make the number of bytes even for the purpose of checksum calculation. The extra byte shall not be transmitted as part of the packet, but its existence shall be assumed at the receiving end for checksum verification.

	Word 1	0001	(Note: all numbers are in hexadecimal)
	Word 2	F203	
	Word 3	F4F5	
	Word 4	F6F7	
Sum showing carry:		2DDF0	
Sum without carry:		DDF0	
	Carry:	2	
Ones-complement sum:		DDF2	
Complement for checksum:		220D	

FIGURE 8. Example of checksumming.

5.2.5 NETBLT protocol operation. Each NETBLT transfer shall have three stages: connection setup, data transfer, and connection close. The stages are described in detail below, along with methods for ensuring that each stage completes reliably. State diagrams are provided at the end of the description for each stage of the transfer. Each transition in the diagrams is labeled with the event that causes the transition, and optionally, in parentheses, actions that occur at the time of the transition.

5.2.5.1 Connection setup. A NETBLT connection shall be set up by an exchange of two packets between the sending NETBLT and the receiving NETBLT. The sending end shall send an OPEN packet; the receiving end shall acknowledge the OPEN packet in one of two ways: it shall either send a REFUSED packet, indicating that the connection cannot be completed for some reason, or it shall complete the connection setup by sending a RESPONSE packet. After a successful connection setup, the transfer can begin. Figure 9 illustrates the opening of a connection by a sender, and figure 10 shows the same process for a receiver.

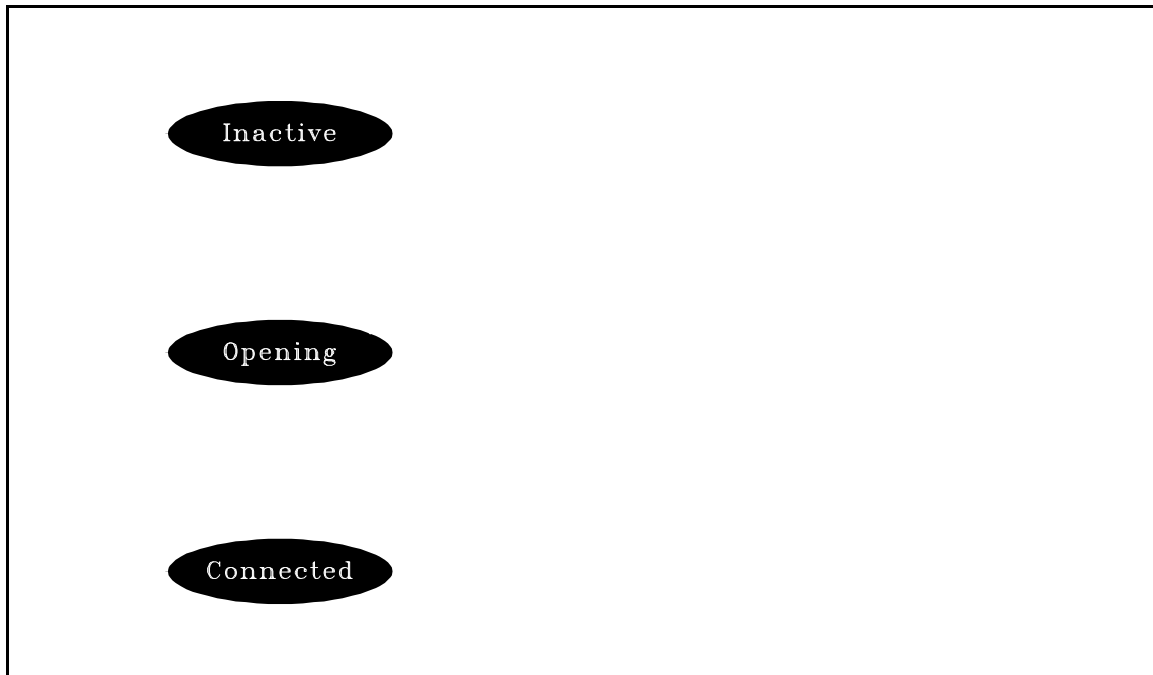


FIGURE 9. Sender open state diagram.

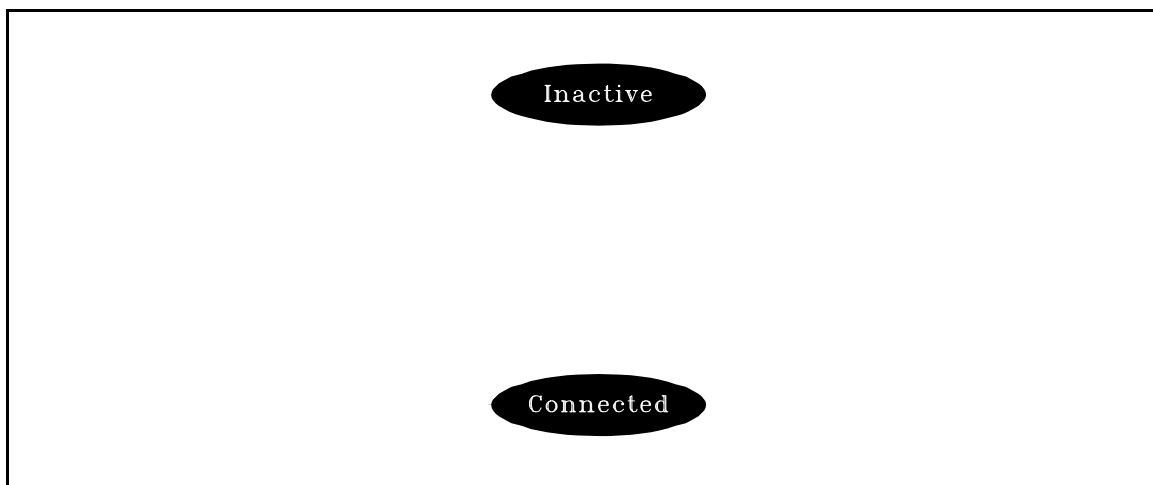


FIGURE 10. Receiver open state diagram.

5.2.5.1.2 Death timeout. Each side of the connection shall transmit its death-timeout value in seconds in the OPEN or the RESPONSE packet. The death-timeout value shall be used to determine the frequency with which to send keepalive packets during idle periods of an opened connection (death timers and keepalive packets are described in 5.2.5.2.5 and 5.2.5.2.6).

5.2.5.1.3 Port number. The sending NETBLT shall specify a passive client through a client-specific "well-known" (see 5.2.9.1.2) 16-bit logical port number on which the receiving end listens. The sending client shall identify itself through a 32-bit Internet address and a locally unique 16-bit port number.

5.2.5.1.4 Client string. An unstructured, variable-length client message field is provided in OPEN and RESPONSE packets for any client-specific information that may be required. In TACO2, this field shall be used to transfer the metamessage provided by the TACO2 NRTS. In addition, a "reason for refusal" field is provided in REFUSED packets.

5.2.5.1.5 OPEN timer. Recovery for lost OPEN and RESPONSE packets shall be provided using timers. The sending end shall set a timer when it sends an OPEN packet. When the timer expires, another OPEN packet shall be sent, until some predetermined maximum number (at least five) of OPEN packets have been sent. The timer shall be cleared upon receipt of a RESPONSE or REFUSED packet.

5.2.5.1.6 Connection ID. To prevent duplication of OPEN and RESPONSE packets, the OPEN packet contains a 32-bit connection unique identifier (UID) that must be returned in the RESPONSE packet. This UID prevents the initiator from confusing the response to the current request with the response to an earlier connection request (there can only be one connection open between any pair of logical ports). Any OPEN or RESPONSE packet with a port pair matching that of an open connection shall have its UID checked. If the UID of the packet matches the UID of the connection, then the packet type shall be checked. If it is a RESPONSE packet, it shall be treated as a duplicate and ignored. If it is an OPEN packet, the passive NETBLT shall send another RESPONSE (on the assumption that a previous RESPONSE packet was sent and lost, causing the initiating NETBLT to retransmit its OPEN packet). A non-matching UID shall be treated as an attempt to open a second connection between the two ports and shall be rejected by sending a REFUSED message.

## 5.2.5.2 Data transfer.

5.2.5.2.1 Single buffer transfer. The simplest full-duplex mode of data transfer shall proceed as follows. The sending client shall set up a buffer full of data. The receiving NETBLT shall send a GO message inside a CONTROL packet to the sender, signifying that it too has set up a buffer and is ready to receive data. Once the GO message is received, the sender shall transmit the buffer as a series of DATA packets followed by an LDATA packet. When the last packet in the buffer should have been received (as determined by a timer, see 5.2.5.2.4.2), if any packets of that buffer have not been received, the receiver shall send a RESEND message inside a CONTROL packet containing a

list of packets that were not received. The sender shall resend these packets. This process shall continue until there are no missing packets. At that time the receiver shall send an OK message inside a CONTROL packet, shall set up another buffer to receive data, and shall send another GO message. The sender, having received the OK message, shall set up another buffer, wait for the GO message, and repeat the process.

5.2.5.2.2 Multiple buffer transfer. A more efficient full-duplex transfer mode uses multiple buffering, in which the sender and receiver allocate and transfer buffers in a manner that allows error recovery or successful transmission confirmation of previous buffers to be concurrent with transmission of the current buffer. During the connection setup phase, one of the negotiated parameters is the number of concurrent buffers permitted during the transfer. If more than one buffer is available, transfer of the next buffer shall start right after the current buffer finishes, and the receiver is ready to receive the buffer. The receiver shall signal that it is ready for the next buffer by sending a GO message. This is illustrated in the following example. Assume the sender has received two buffers, A and B, in a multiple-buffer transfer, with A preceding B. When A has been transferred and the sending NETBLT is waiting for either an OK or a RESEND message for it, the sending NETBLT can start sending B immediately,. If the receiver of data sends an OK for A, all is well; if it receives a RESEND, the missing packets specified in the RESEND message shall be retransmitted. In the multiple-buffer transfer mode, all packets to be sent shall be ordered by buffer number (lowest number first). Since buffer numbers increase monotonically, packets from an earlier buffer shall always precede packets from a later buffer.

#### 5.2.5.2.3 Recovering from lost control messages.

5.2.5.2.3.1 Control packet. NETBLT shall use a single long-lived control packet; the packet shall be treated like a First-In-First-Out queue, with new control messages added on at the end and acknowledged control messages removed from the front. The implementation shall place control messages in the control packet and shall transmit the entire control packet, consisting of any unacknowledged control messages plus new messages just added. The entire control packet shall also be transmitted whenever the control timer expires. Since control packet transmissions are fairly frequent, unacknowledged messages may be transmitted several times before they finally are acknowledged. The receiver may send zero or more control messages (OK, GO, or RESEND) within a single CONTROL packet.

5.2.5.2.3.2 Control timer. Whenever the receiver sends a control packet, it shall start a control timer. When the control timer expires, the receiving NETBLT shall resend the control packet and reset the timer. The receiving NETBLT shall continue to resend control packets in response to control timer expiration until either the control timer is cleared or the receiving NETBLT's death timer (described in 5.2.5.2.5) expires (at which time it shall shut down the connection). The appropriate initial control timer value is constrained by the characteristics of the link. Subsequent control packets shall have their timer values based on the network round-trip transit time (the time between sending the control packet and receiving the acknowledgment of all messages in the control packet) plus a variance factor. The timer value shall be updated continually, based on a smoothed average of collected round-trip transit times. The control timer shall be set to the keepalive value when a packet is received from the sender with high-acknowledged-sequence-number equal to the

highest sequence number in the control packet most recently sent (see 5.2.5.2.3.3). The current value of the control timer is provided to the sender in each OK message.

NOTE: The exact algorithm for control timer calculation is not a required part of this standard. The recommended algorithm is as follows:

- a. Initially, the round trip time is set to the keepalive value and the deviation is set to zero.
- b. Thereafter, new smoothed round trip time =  $(1-a) * \text{old smoothed round trip time} + a * \text{latest round trip measurement}$
- c. New deviation =  $(1-b) * \text{old deviation} + b * |\text{latest round trip measurement} - \text{old smoothed round trip time}|$

where  $a = 1/8$  and  $b = 1/4$ , allowing computations to be done with add and shift operations. The control timer is set equal to the new smoothed round trip time plus twice the new deviation, or to the keepalive value, whichever is less, if the control packet is not empty. If the control packet is empty, the control timer is set to the keepalive value.

**5.2.5.2.3.3 Control message sequence number.** Each control message shall include a sequence number, which starts at one and increases by one for each control message sent. The sending NETBLT shall check the sequence number of every incoming control message against all other sequence numbers it has received. It shall store the highest sequence number below which all other received sequence numbers are consecutive (in following paragraphs this is called the high-acknowledged-sequence-number) and shall return this number in every packet flowing back to the receiver. The receiver shall remove control messages with sequence numbers less than or equal to the high-acknowledged-sequence-number from the control packet. The receiver shall set its control timer to the keepalive value (see 5.2.5.2.6) when it receives a packet from the sender with a high-acknowledged-sequence-number equal to the highest sequence number in the control packet just sent.

**5.2.5.2.3.4 Response to control messages.** The sending NETBLT, upon receiving a CONTROL packet, shall either set up a new buffer (upon receipt of an OK message for a previous buffer), mark data for resending (upon receipt of a RESEND message), or prepare a buffer for sending (upon receipt of a GO message). If the sending NETBLT is not in a position to send data, it shall send a NULL-ACK packet, which contains its high-acknowledged-sequence-number (this permits the receiving NETBLT to acknowledge any outstanding control messages), and wait until it can send more data.

**5.2.5.2.4 Recovering from lost DATA and LDATA packets.** NETBLT solves the problem of DATA and LDATA packet loss by using a data timer for each buffer at the receiving end. The simplest data timer model has a data timer set when a buffer is ready to be received; if the data timer expires, the receiving NETBLT shall send a RESEND message requesting all missing DATA/LDATA packets in the buffer. When all packets have been received, the timer shall be cleared. Data timer values shall be based on the amount of time taken to transfer a buffer (as determined by the number of DATA packet bursts in the buffer times the burst interval) plus a variance factor.

5.2.5.2.4.1 Send buffer state sequence. The state sequence for a sending buffer shall be as follows: when a GO message for the buffer is received, the buffer is created, filled with data, and placed in a SENDING state. When an OK for that buffer has been received, it goes into a SENT state and may be released. Figure 11 illustrates this sequence.

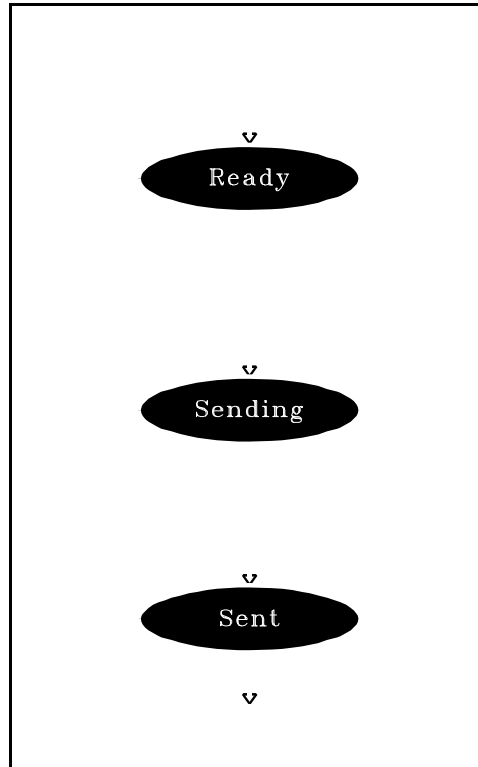


FIGURE 11. Sending buffer state diagram.

5.2.5.2.4.2 Receive buffer state sequence. The state sequence for a receiving buffer is a little more complicated. Assume existence of Buffer A. When a control message for Buffer A is sent, the buffer shall move into state ACK-WAIT (it is waiting for acknowledgement of the control message). As soon as the control message has been acknowledged, Buffer A shall move from the ACK-WAIT state into the ACKED state (it is now waiting for DATA packets to arrive). At this point, the control message shall be removed from the control packet. Buffer A shall stay in the ACKED state until a DATA, LDATA, or NULL-ACK packet arrives with its "Last Buffer Touched" number greater than or equal to Buffer A's number. At this time, Buffer A's data timer shall be set to the time expected for the remaining packets in the buffer to be received plus a variance, and Buffer A shall move to the RECEIVING state. (Note: This mechanism is different from, and simpler than, the "loose/tight" timer mechanism described in Request for Comment (RFC) 998). When all DATA packets for A have been received, it shall move from the RECEIVING state to the RECEIVED state and may be passed to the receiving client. Had any packets been missing, Buffer A's data timer would have expired; in that case, Buffer A shall move into the ACK-WAIT state after sending a RESEND message. The sending of a RESEND message shall cause the data timers of all buffers currently in the RECEIVING

state to be recalculated, since the presence of resend packets will change the expected completion time for later buffers. The state progression would then move as in the above example. Figure 12 illustrates this sequence.

NOTE: The exact algorithm for data timer estimation is not a required part of this standard. The recommended algorithm is to compute the number of packets expected before the buffer is complete, multiply that by the time required to transmit a packet, and add a variance of 50 percent. The time required to transmit a packet is the burst interval divided by the burst size.

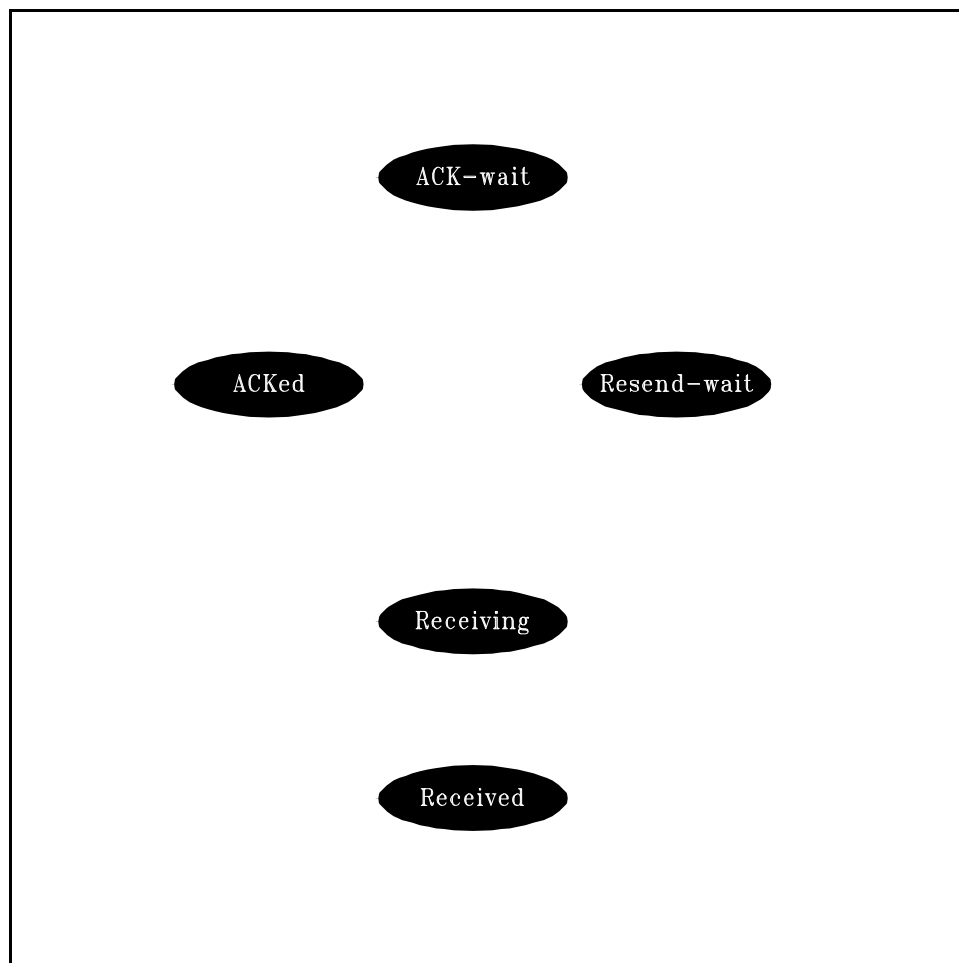


FIGURE 12. Receiving buffer state diagram.

5.2.5.2.4.3 Alternative method for data timer estimation. When the burst interval is set to zero, an alternative method for estimating the time required to send a packet shall be used for the purpose of data timer estimation. The alternative algorithm is to measure the elapsed time to receive a consecutively-numbered sequence of packets, and to divide that time by the number of packets in the sequence to calculate the time required for transmission of a single packet. This value may then

be used in the algorithm recommended in 5.2.5.2.4.2. This value may be smoothed, in a manner consistent with that recommended for control timer calculation in 5.2.5.2.3.2. A suggested initial value used by the receiver for the time required to transmit a single packet is  $(\text{sender's death timeout value} * \text{packet size}) / (\text{buffer size} * \text{maximum outstanding buffers} * 4)$ .

**5.2.5.2.5 Death timers.** At connection startup, each NETBLT shall send its death value to the other end in the OPEN or the RESPONSE packet. As soon as the connection is opened, each end shall set its death timer to its chosen value; this timer shall be reset every time a packet is received. When a NETBLT's death timer expires, it shall close the connection without sending any more packets. A recommended value for the death timer is in 5.2.9.10.

**5.2.5.2.6 Keepalive packets.** NETBLT shall include a keepalive function, which sends packets repeatedly at fixed intervals when a NETBLT has no other reason to send packets. The sender shall use NULL-ACKs as keepalive packets; the receiver shall use empty CONTROL packets. If the sending NETBLT is not ready to send upon receipt of a control packet, it shall send a single NULL-ACK packet to clear any outstanding control timers at the receiving end. Each end shall use the other end's death-timeout value to compute a frequency with which to send keepalive packets. The keepalive frequency shall be high enough that several keepalive packets can be lost before the other end's death timer expires; recommended values are the sender's death timer value divided by seven for the receiver, and the receiver's death timer value divided by eight for the sender. Keepalive intervals should be different to avoid repeated collisions in half-duplex operations.

**5.2.5.3 Terminating the connection.** The four conditions under which a connection shall be terminated are a successful transfer, a client quit or abort, a NETBLT abort, and a death timer timeout.

**5.2.5.3.1 Successful transfer** After a successful data transfer, NETBLT shall close the connection.

**5.2.5.3.1.1 Receiver successful close.** When the sender is transmitting the last buffer of data, it shall set a "last-buffer" flag on every DATA packet in the buffer. The receiver shall recognize that the transfer has completed successfully when all of the following are true: (1) it has received DATA packets with a "last-buffer" flag set, (2) all its control messages have been acknowledged, and (3) it has no outstanding buffers with missing packets. The DONE packet shall be transmitted when the receiver recognizes that the transfer has been completed successfully. At that point, the receiver shall close its half of the connection. Figure 13 illustrates this sequence.

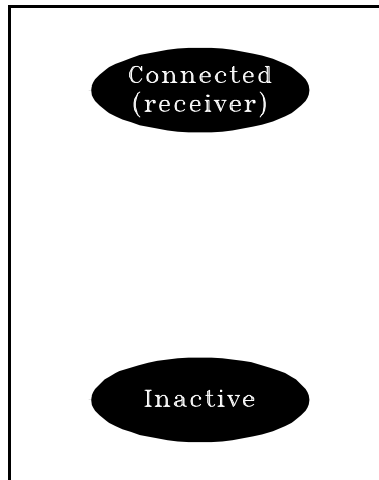


FIGURE 13. Receiver successful close state diagram.

5.2.5.3.1.2 Sender successful close. The sender shall recognize that the transfer has completed when the following are true: (1) it has transmitted DATA packets with a "last-buffer" flag set and (2) it has received OK messages for all its buffers. At that point, it shall "dally" for a predetermined period of time before closing its half of the connection. If the NULL-ACK packet acknowledging the receiver's last OK message was lost, the receiver has time to retransmit the OK message, receive a new NULL-ACK, and recognize a successful transfer. The dally timer value shall be based on the receiver's control timer value; it shall be long enough to allow the receiver's control timer to expire so that the OK message can be resent. The sender shall use the receiver's current control timer value to compute its dally timer value. A value of twice the receiver's control timer value is suitable for the dally timer. When the sender receives a DONE packet, it shall clear its dally timer and close its half of the connection. Figure 14 illustrates this sequence.



FIGURE 14. Sender successful close state diagram.

5.2.5.3.2 Client QUIT. During a NETBLT transfer, one client may send a QUIT packet to

the other, to terminate the transfer prematurely. Since the QUIT occurs at a client level, the QUIT transmission shall occur only between buffer transmissions. The NETBLT receiving the QUIT packet shall take no action other than immediately notifying its client and transmitting a QUITACK packet. The QUIT sender shall time out and retransmit until a QUITACK has been received or its death timer expires. The sender of the QUITACK shall dally before quitting, so that it can respond to a retransmitted QUIT. Figure 15 illustrates this sequence.

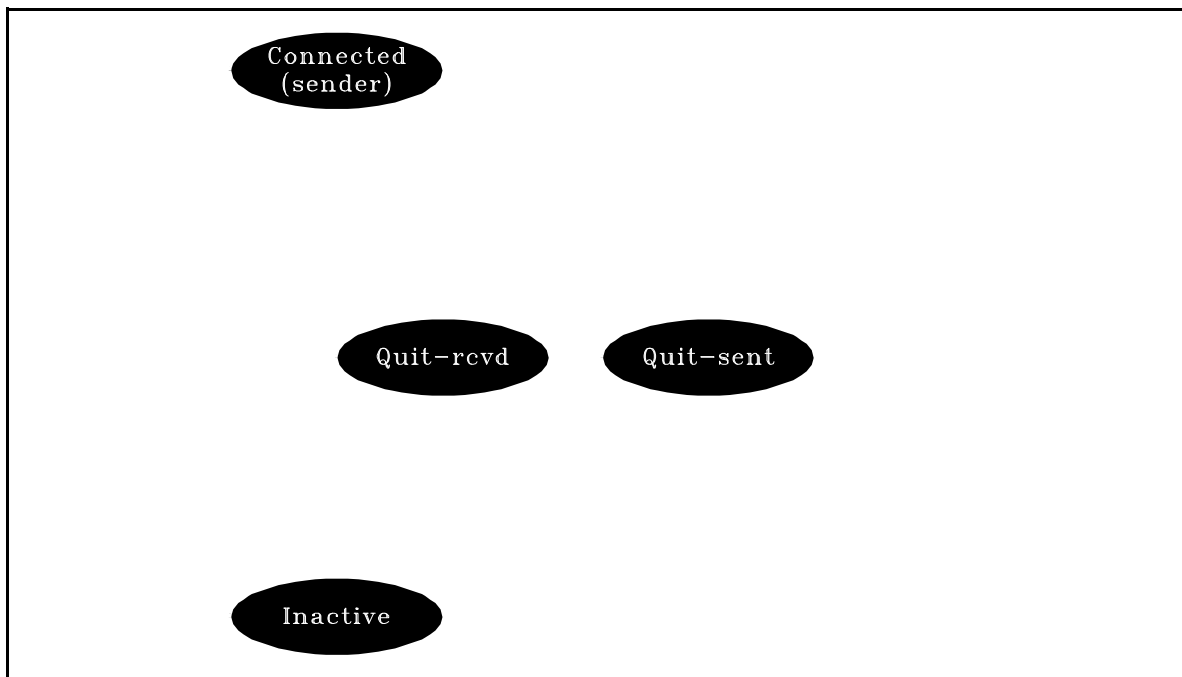


FIGURE 15. Quit state diagram.

5.2.5.3.3 NETBLT ABORT. An ABORT shall take place when an unrecoverable malfunction occurs. Since the ABORT originates in the NETBLT layer, it may be sent at any time. The ABORT implies that the NETBLT layer is malfunctioning, so no transmit reliability is expected, and the sender shall immediately close its connection. Figure 16 illustrates this sequence.

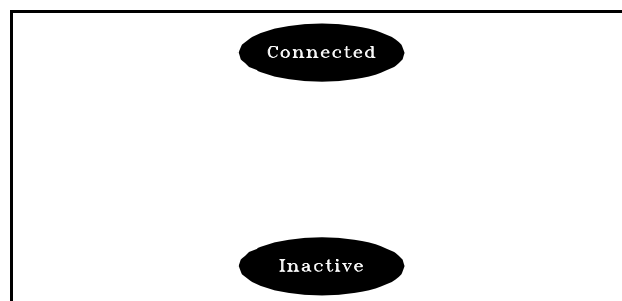


FIGURE 16. Abort state diagram.

5.2.5.3.4 Death timer timeout. When a NETBLT's death timer expires, it shall close the connection without sending further packets.

5.2.6 Protocol layering structure. NETBLT shall be implemented directly on top of the IP. It has been assigned an official protocol number of 30 (decimal), which is 0x1E (hexadecimal).

5.2.7 Packet formats. NETBLT packet formats used in TACO2 shall be in accordance with those given in this section. NETBLT packets are divided into three categories, all of which share a common 12-byte packet header.

- a. There are three packet types that travel only from data sender to receiver; these contain the high-acknowledged-sequence-numbers which the receiver uses for control message transmission reliability. These are the NULL-ACK, DATA, and LDATA packets.
- b. There is one packet type that travels only from receiver to sender. This is the CONTROL packet. Each CONTROL packet can contain an arbitrary number of control messages (GO, OK, or RESEND), each with its own sequence number.
- c. There are seven packet types which can travel in either direction, although the TACO2 usage of NETBLT may limit their direction of travel. These packet types either have special ways of insuring reliability, or are not transmitted reliably. They are the OPEN, RESPONSE, REFUSED, QUIT, QUITACK, DONE, and ABORT packets. In the TACO2 usage of NETBLT, the OPEN packet shall travel from sender to receiver; the RESPONSE, REFUSED, and DONE packets shall travel from receiver to sender; and the QUIT, QUITACK, and ABORT packets can be sent by both sending and receiving NETBLTs.

All packet headers shall be "longword-aligned," such as, all packet headers are a multiple of four bytes in length and all four-byte fields start on a longword boundary. The Client String field shall be terminated with at least one null byte, with extra null bytes added at the end to create a field that is a multiple of four bytes long. All numeric values shall be coded as binary integers.

#### 5.2.7.1 OPEN (type 0) and RESPONSE (type 1).

1)

- a. Checksum: To generate the checksum, the checksum field itself is cleared, the 16-bit ones-complement sum is computed over the packet, and the ones complement of this sum is placed in the checksum field.
- b. Version: The NETBLT protocol version number
- c. Type: The NETBLT packet type number (OPEN = 0, RESPONSE =
- d. Length: The total length (NETBLT header plus data, if present) of the NETBLT packet in bytes
- e. Local Port: The local NETBLT's 16-bit port number
- f. Foreign Port: The foreign NETBLT's 16-bit port number
- g. Connection UID: The 32 bit connection UID specified in 5.2.5.1.6.
- h. Buffer size: Size in bytes of each NETBLT buffer (except the last)
- i. Data packet size: Length of each DATA packet in bytes
- j. Burst Size: Number of DATA packets in a burst
- k. Burst Interval: Transmit time in milliseconds of a single burst

- l. Death timer: Packet sender's death timer value in seconds
- m. "C": The DATA packet data checksum flag (0 = do not checksum DATA packet data, 1 = do). Shall be 1 in TACO2.
- n. "M": The transfer mode (0 = READ, 1 = WRITE). (see 5.2.9.2.5).
- o. Maximum # Outstanding Buffers: Maximum number of buffers that can be transferred before waiting for an OK message from the receiving NETBLT.
- p. Client string: An arbitrary, null-terminated, longword-aligned string for use by NETBLT clients. Contains the metamessage in TACO2.

(NOTE: the Reserved (MBZ) field may be used as the Connection Number field by the Header Abbreviation sublayer (see 5.4.1)).

#### 5.2.7.2 QUITACK (type 3), and DONE (type 10).

#### 5.2.7.3 QUIT (type 2), ABORT (type 4), and REFUSED (type 9).

#### 5.2.7.4 DATA (type 5) and LDATA (type 6).

- far.
- a. Checksum: Checksum of the packet header only, including the Data Area Checksum Value.
  - b. Buffer number: A 32 bit unique number assigned to every buffer. Numbers are monotonically increasing, starting with 1.
  - c. Last Buffer Touched: The number of the highest buffer transmitted so far.
  - d. High Consecutive Sequence Number Received: Highest control message sequence number below which all control message sequence numbers received are consecutive.
  - e. Packet number: Monotonically increasing DATA packet identifier, starting with zero in each buffer.
  - f. Data Area Checksum Value: Checksum of the DATA packet's data. Algorithm used is the same as that used to compute checksums of other NETBLT packets.
  - g. "L" is a bit that is set to 1 when the buffer that this DATA packet belongs to is the last buffer in the transfer.
  - h. New Burst Size: Burst size as negotiated from value given by receiving NETBLT in OK message.

- i. New Burst Interval: Burst interval as negotiated from value given by receiving NETBLT in OK message. Value is in milliseconds.

#### 5.2.7.5 NULL-ACK (type 7).

- a. Last Buffer Touched: The number of the highest buffer transmitted so far.
- b. High Consecutive Sequence Number Received: Same as in DATA/LDATA packet.
- c. New Burst Size: Burst size as negotiated (half- and full-duplex only) from value given by receiving NETBLT in OK message.
- d. New Burst Interval: Burst interval as negotiated (half- and full-duplex only) from value given by receiving NETBLT in OK message. Value is in milliseconds.
- e. "L" is a bit that is set to 1 when the buffer identified in the Last Buffer Touched field is the last buffer in the transfer.

#### 5.2.7.6 CONTROL (type 8).

Followed by any number of messages, each of which is longword aligned, with the following formats:

5.2.7.6.1 GO message (type 0).

- a. Type: Message type (GO = 0, OK = 1, RESEND = 2)
- b. Sequence number: A 16-bit unique message number. Sequence numbers must be monotonically increasing, starting with 1.
- c. Buffer number: As in DATA/LDATA packet

5.2.7.6.2 OK message (type 1).

- a. New offered burst size: Burst size for subsequent buffer transfers, possibly based on performance information for previous buffer transfers.
- b. New offered burst interval: Burst interval for subsequent buffer transfers, possibly based on performance information for previous buffer transfers. Interval is in milliseconds.
- c. Current control timer value: Receiving NETBLT's control timer value in milliseconds.

#### 5.2.7.6.3 RESEND message (type 2).

- a. Packet number: The 16-bit data packet identifier of a DATA packet, from the buffer identified by Buffer Number, whose retransmission is requested. Multiple packet numbers may occur in one RESEND message.

5.2.8 Required NETBLT components. TACO2 uses three modes of NETBLT operation; simplex, half-duplex, and full-duplex. This section identifies the required components of NETBLT for each mode of operation.

5.2.8.1 Simplex. The only NETBLT packet types used in the simplex case shall be the following:

- a. OPEN
- b. QUIT
- c. ABORT
- d. DATA
- e. LDATA
- f. NULL-ACK

5.2.8.1.1 Sender simplex operation. Operation of NETBLT in simplex send mode shall be as follows: the OPEN message is sent; DATA, LDATA, and possibly NULL-ACK packets are sent; and the connection is closed. Any packet may be sent more than once, for redundancy, but for all n, packets from buffer(n - 1) shall not be sent after packets from buffer(n). QUIT and ABORT packets may be sent at any time, and shall have the same effect. The Maximum Number of Outstanding

Buffers (in the OPEN packet) shall be set to 2.

5.2.8.1.2 Receiver simplex operation. Operation of NETBLT in simplex receive mode shall be as follows: when an OPEN packet is received, a connection is considered to be established. Packets received shall be stored into NETBLT BUFFERS. The receiving NETBLT shall pass a buffer to the client when the buffer is filled with correct packets or when good packets for a higher-numbered buffer are received. A list of packets that are possibly bad, or missing, shall be passed to the client. When the last buffer (L flag set in packet headers) has been passed to the client, or when the death timeout has expired, the receiving connection shall be terminated.

5.2.8.1.2.1 Packet error handling. The receiving NETBLT shall discard redundant packets. In the case of errors, the following rules shall apply at the receiving NETBLT:

- a. A NETBLT packet with a bad checksum shall be discarded, unless it is a DATA or LDATA packet.
- b. A NETBLT DATA or LDATA packet, with a bad header checksum or data area checksum, optionally may be saved but flagged as possibly bad. Reasonableness checks shall be used to insure that good data is not affected by the possibly bad packet header or data. If a good NETBLT packet (redundantly transmitted) is received with the same buffer and packet number as a possibly bad one, the possibly bad packet shall be replaced with the good one.

5.2.8.2 Half-duplex. The normal, full-duplex version of NETBLT shall operate across half-duplex connections with the following modification: keepalive packets shall not be sent by the receiver while it is in the process of receiving a packet. The burst timer and burst size counter shall be reset at the start of each transmission period. If the Maximum Number of Outstanding Buffers (in the OPEN packet) is set to 1, the sending and receiving NETBLTs will operate in lockstep. If the Maximum Number of Outstanding Buffers is set to a value N greater than 1, the receiving NETBLT shall wait until N buffers have been completely received or have had their data timers expire before sending a CONTROL packet. An exception occurs when the last buffer is sent; when all buffers up to and including the last buffer have been completely received or have had their data timers expire, the receiving NETBLT shall be permitted to send its CONTROL packet. The last buffer is identified by the receiver as the buffer for which the "L" bit is set in a DATA/LDATA packet, or as the Last Buffer Touched in a NULL-ACK packet with its "L" bit set to 1.

NOTE: Early implementations of TACO2 did not include the "L" bit in the NULL-ACK packet. These implementations must set the Maximum Number of Outstanding Packets to 1; otherwise, loss of the entire last buffer, due to communications error, would cause the receiver to stop operating.

5.2.8.3 Full-duplex. Across full-duplex connections the normal NETBLT as described in 5.2.1 through 5.2.6 shall be used.

5.2.9 Specific values for NETBLT. The following are comments on, or specific values for,

various NETBLT fields.

#### 5.2.9.1 Fields common to all packets.

##### 5.2.9.1.1 Version. TACO2 shall use version 4 of NETBLT.

5.2.9.1.2 Local port and foreign port. The "well-known" port, to which OPEN messages should be directed, shall be port number 1. This signifies that a connection with the TACO2 NRTS is to be established. Some other randomly-selected value shall be used for local port, this value shall be unique in that if more than one NETBLT connection is supported by a single host interface, the port number shall not be duplicated.

##### 5.2.9.1.3 Longword alignment padding. The content of these fields shall be zeros.

#### 5.2.9.2 OPEN and RESPONSE packets.

5.2.9.2.1 Connection UID. Connection UID may be any randomly-selected value, which shall be unique in that if more than one NETBLT connection is supported by a single host interface, it shall not be duplicated. More than one connection may be supported concurrently, to support multiplexing, but only the ability to support a single connection is required.

5.2.9.2.2 Buffer size. A TACO2 implementation shall support a buffer size of at least 4096 data bytes. In addition, it shall support any number of DATA/LDATA packets per buffer up to and including 32 packets; support for a larger number of packets per buffer is allowed but not required.

5.2.9.2.3 DATA packet size. A TACO2 implementation shall support a maximum DATA and LDATA packet size of at least 512 data bytes. DATA packets as small as 64 data bytes shall be supported. LDATA packets may be as small as one data byte, depending on the relationship between message, buffer, and packet sizes.

5.2.9.2.4 Burst size and burst interval. In point-to-point connections, burst size and burst interval ordinarily shall be set to produce the maximum data flow the connection and the hosts can support (such as,  $[\text{burst\_size} * \text{bytes\_per\_packet} * \text{bits\_per\_byte} * 1000 / \text{burst\_interval}]$  shall be approximately equal to the apparent bits per second sent to the link). Alternatively, the burst interval may be set to zero, in which case no internal flow control shall be imposed (see 5.2.3.5).

5.2.9.2.5 Direction. (Effectivity 4) Until the effectivity date, operation of TACO2 is defined only for "M" set to 1; that is, TACO2 allows only active sending and passive receiving. Following that date, operation with "M" set to 0 is also permissible.

5.2.9.2.6 Checksumming. The value of "C" bit shall be 1; that is, TACO2 requires data checksums on DATA/LDATA packets.

5.2.9.2.7 Maximum number of outstanding buffers. A TACO2 implementation shall be capable of supporting at least two concurrently outstanding buffers.

5.2.9.2.8 Client string. This field shall contain the metamessage, which is generated by the TACO2 NRTS. Both OPEN and RESPONSE messages shall include the metamessage; a mechanism for negotiation of the value of some components is defined in the TACO2 NRTS.

#### 5.2.9.3 QUIT packets.

5.2.9.3.1 Reason for QUIT/ABORT/REFUSE. The reason shall be an appropriate ASCII string up to 80 characters long, suitable for display to the recipient. The use of QUIT implies preemption or a probable malfunction.

(NOTE: Strings used may include:

- a. "Error: unknown return from setup upcall"
- b. "Error: fatal application buffer setup error"
- c. "Error: unknown return from flush upcall"
- d. "Error: fatal application buffer flush error"
- e. "Error: fatal buffer setup error"
- f. "Error: unknown return from buffer flush upcall")

#### 5.2.9.4 ABORT packets.

5.2.9.4.1 Reason for QUIT/ABORT/REFUSE. The reason shall be an appropriate ASCII string up to 80 characters long, suitable for display to the recipient. The use of ABORT implies a serious malfunction.

(NOTE: Strings used may include:

- a. "fatal buffer create error"
- b. "test request"
- c. "received strange burst size"
- d. "received strange burst interval"
- e. "Control packet buffer overflow"
- f. "no memory for outbound-packet source route"
- g. "attempt to establish > 1 connection per port pair")

#### 5.2.9.5 REFUSED packets.

5.2.9.5.1 Reason for QUIT/ABORT/REFUSE. The reason shall be an appropriate ASCII string up to 80 characters long, suitable for display to the recipient. The use of REFUSED indicates that the connection cannot be completed for some reason.

(NOTE: Strings used may include:

"no service listening on port x," where x is the unacceptable port number.)

5.2.9.6 DATA and LDATA packets.

5.2.9.6.1 Packet number. The first data packet in each buffer shall be numbered 0.

5.2.9.6.2 Data area checksum value. All TACO2 DATA and LDATA packets shall be checksummed.

5.2.9.7 Timer precision. Timer precision in NETBLT shall be no worse than  $\pm 100$  milliseconds (msec).

5.2.9.8 Open timer value. The open timer shall initially be set to no less than two seconds. In half-duplex and full duplex, the value of the open timer shall be increased by two seconds after each timeout.

5.2.9.9 Quit timer value. The quit timer shall be set to no less than five seconds.

5.2.9.10 Death timer value. The death timer should be set to no less than two minutes.